682 Midterm Review

Max Hamilton and Oindrila Saha

Midterm Review

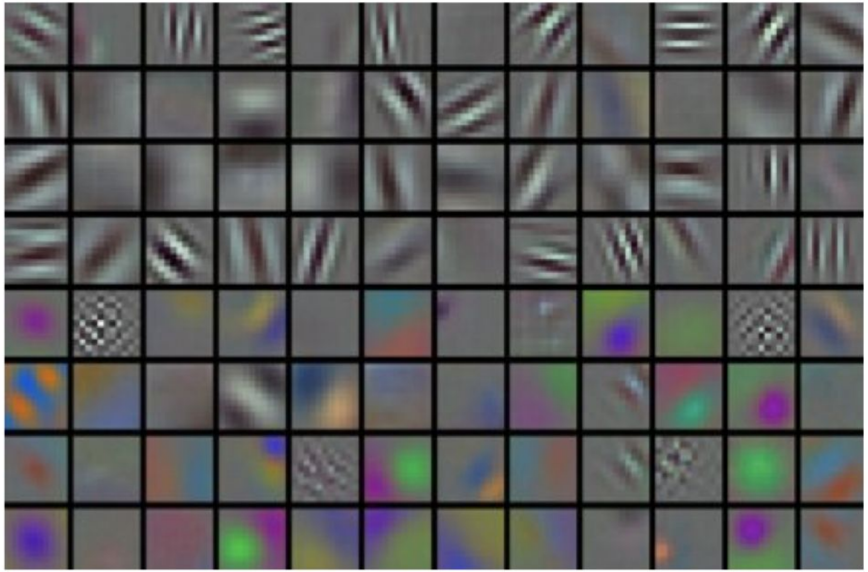
Today we will cover:

- Visualizing ConvNets
- Adversarial Training
- Style Transfer
- CNNs for spatial tasks: detection, segmentation

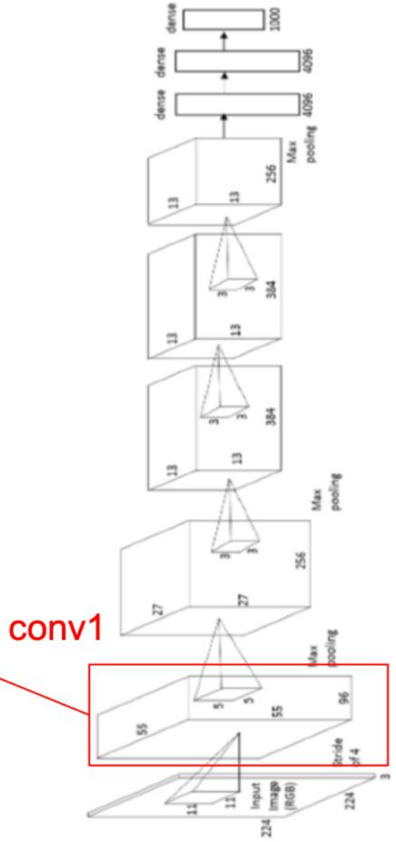
See Review Guide on piazza for full list of topics

Visualizing ConvNets

Visualize the filters/kernels (raw weights)



one-stream AlexNet



only interpretable on the first layer :(

Visualize patches that maximally activate neurons

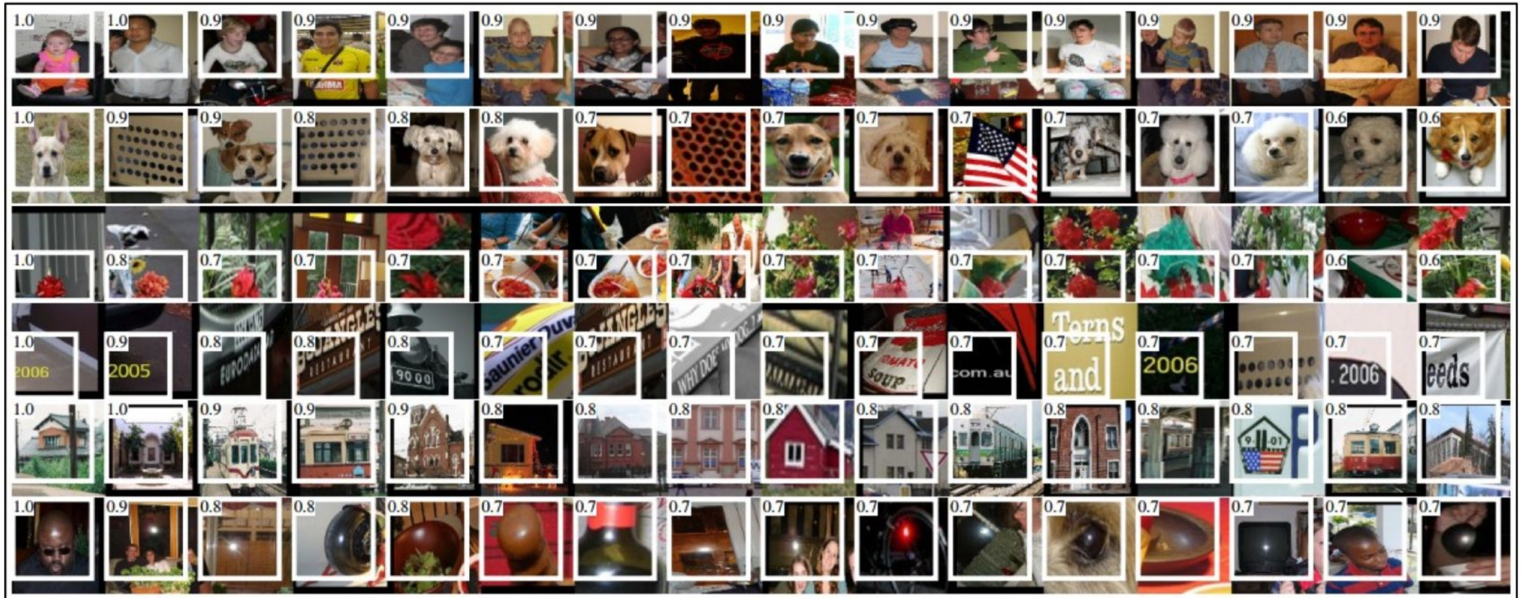
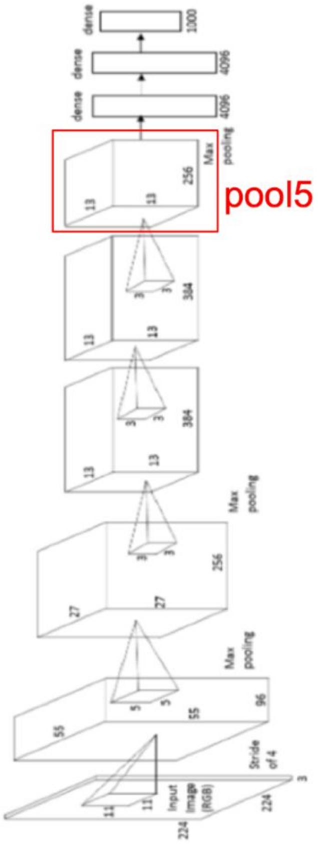


Figure 4: Top regions for six $pool_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

one-stream AlexNet



Rich feature hierarchies for accurate object detection and semantic segmentation
[Girshick, Donahue, Darrell, Malik]

Visualizing the representation

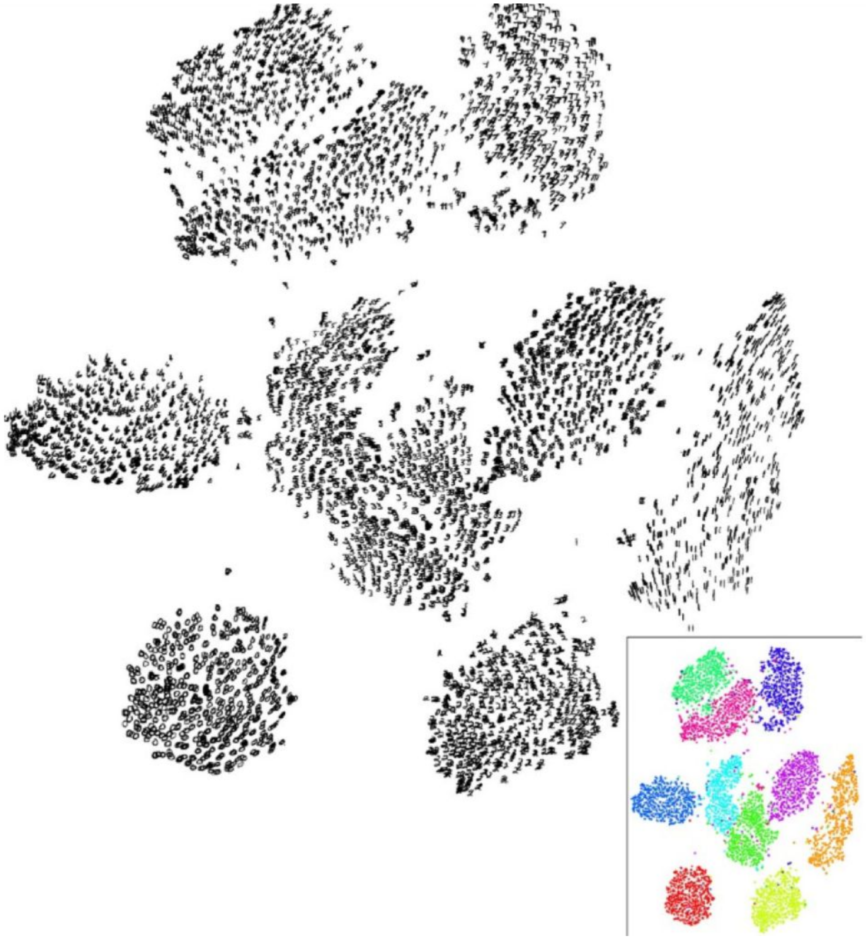
t-SNE visualization

[van der Maaten & Hinton]
(t-distributed stochastic neighbor embed.)

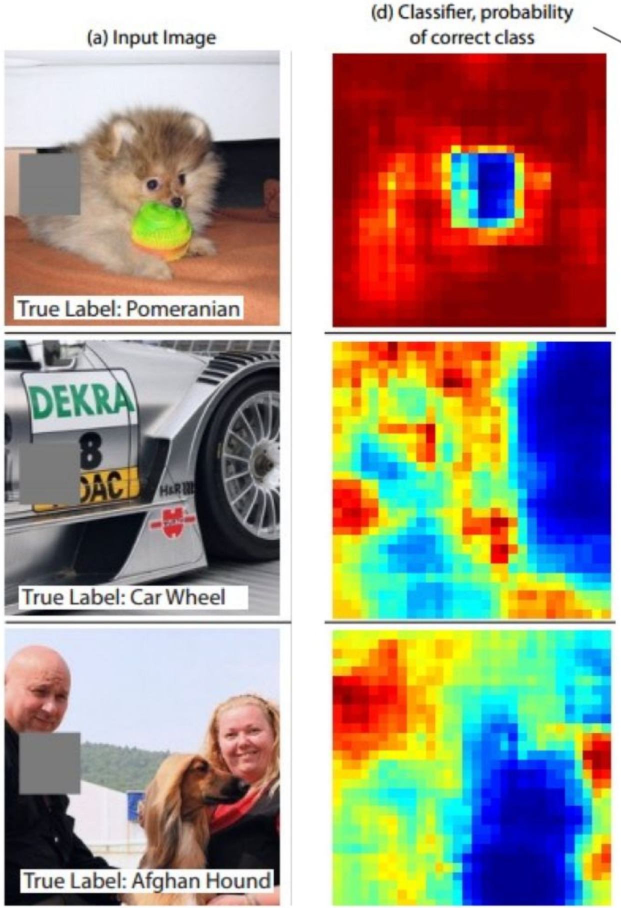
Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places. dissimilar things end up wherever

Right: Example embedding of MNIST digits (0-9) in 2D



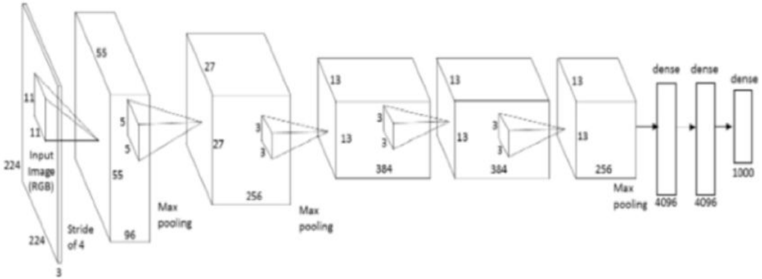
Occlusion experiments
[Zeiler & Fergus 2013]



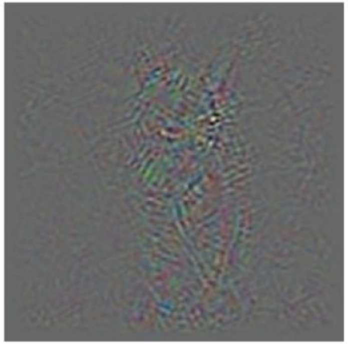
(as a function of the position of the square of zeros in the original image)

Deconv approaches

1. Feed image into net

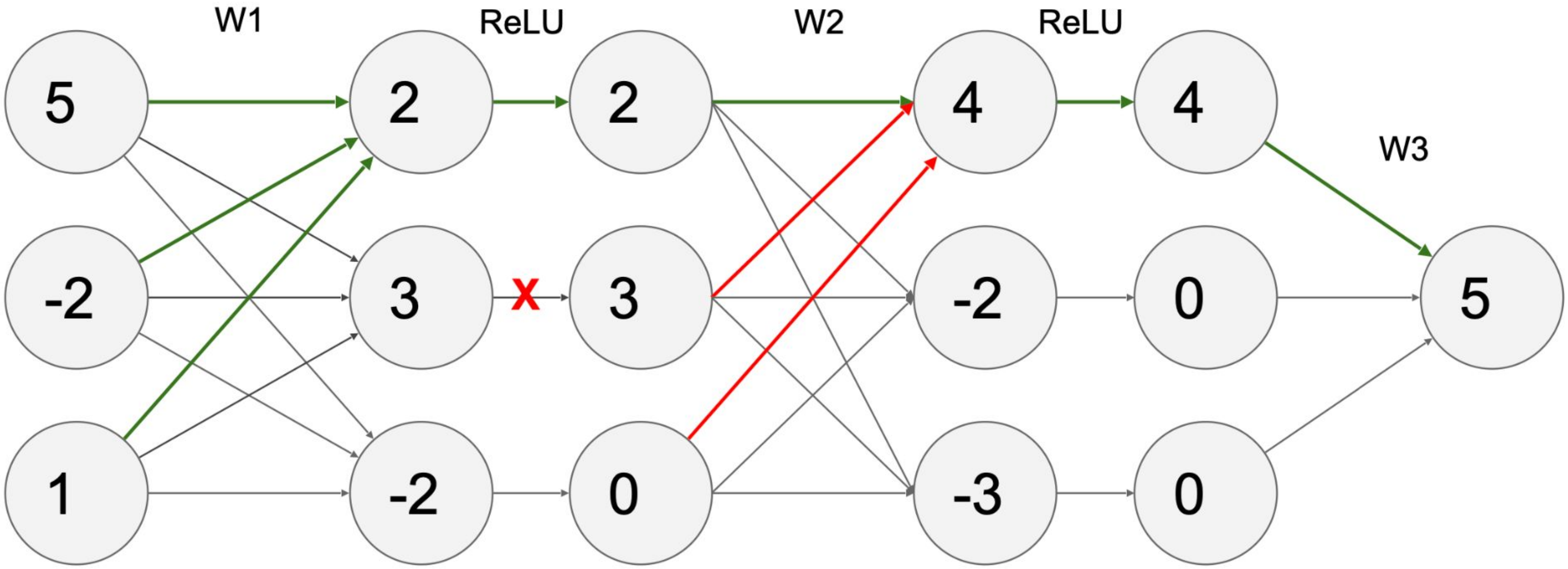


2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:



“Guided backpropagation:” instead

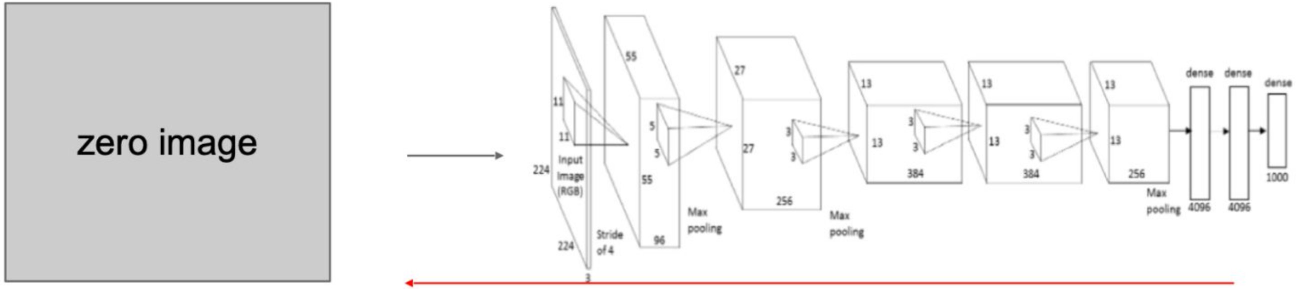
In guided backprop: cancel out -ve paths of influence at each step
(i.e. we only keep positive paths of influence)



positive gradient, negative gradient, zero gradient

Optimization to Image

1. feed in zeros.

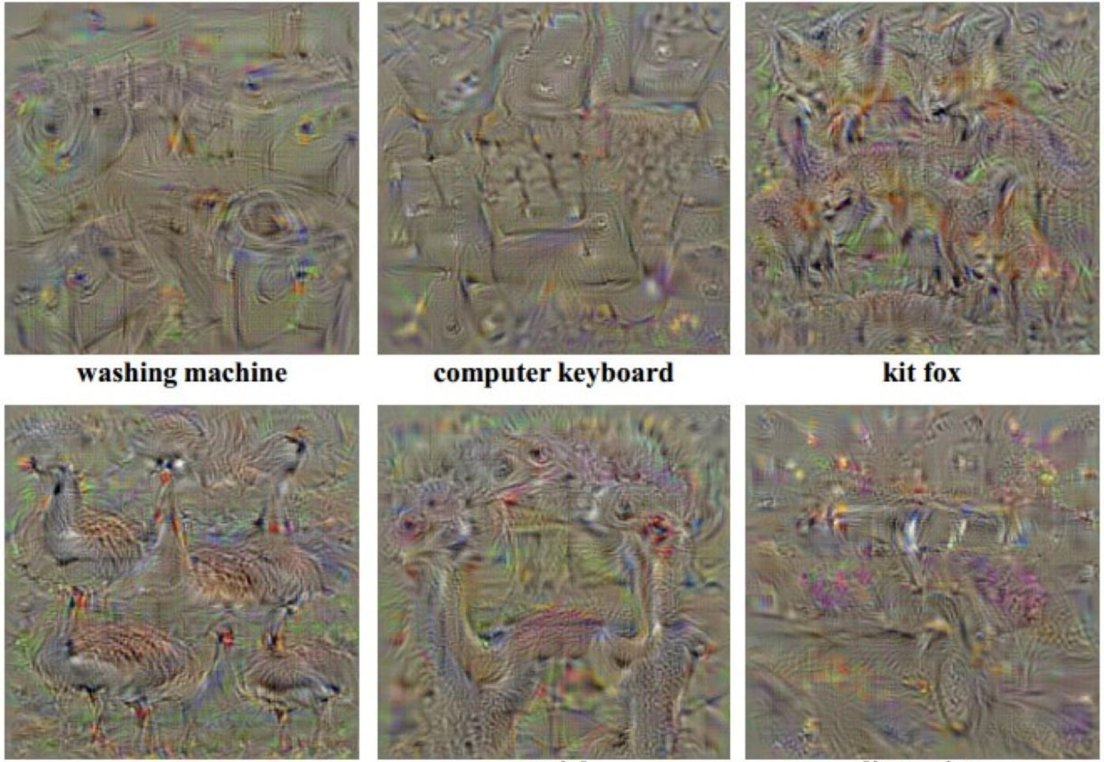


2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image
3. do a small “image update”
4. forward the image through the network.
5. go back to 2.

$$\boxed{\arg\max_{I} \boxed{S_c(I)} - \lambda \|I\|_2^2}$$

score for class c (before Softmax)

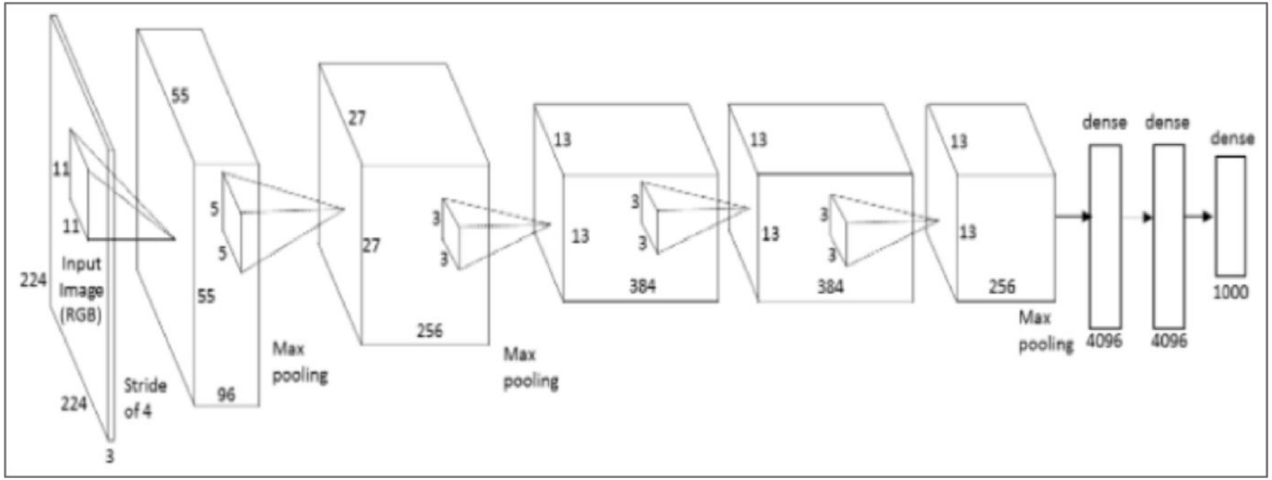
Find images that maximize some class score:



washing machine | computer keyboard | kit fox

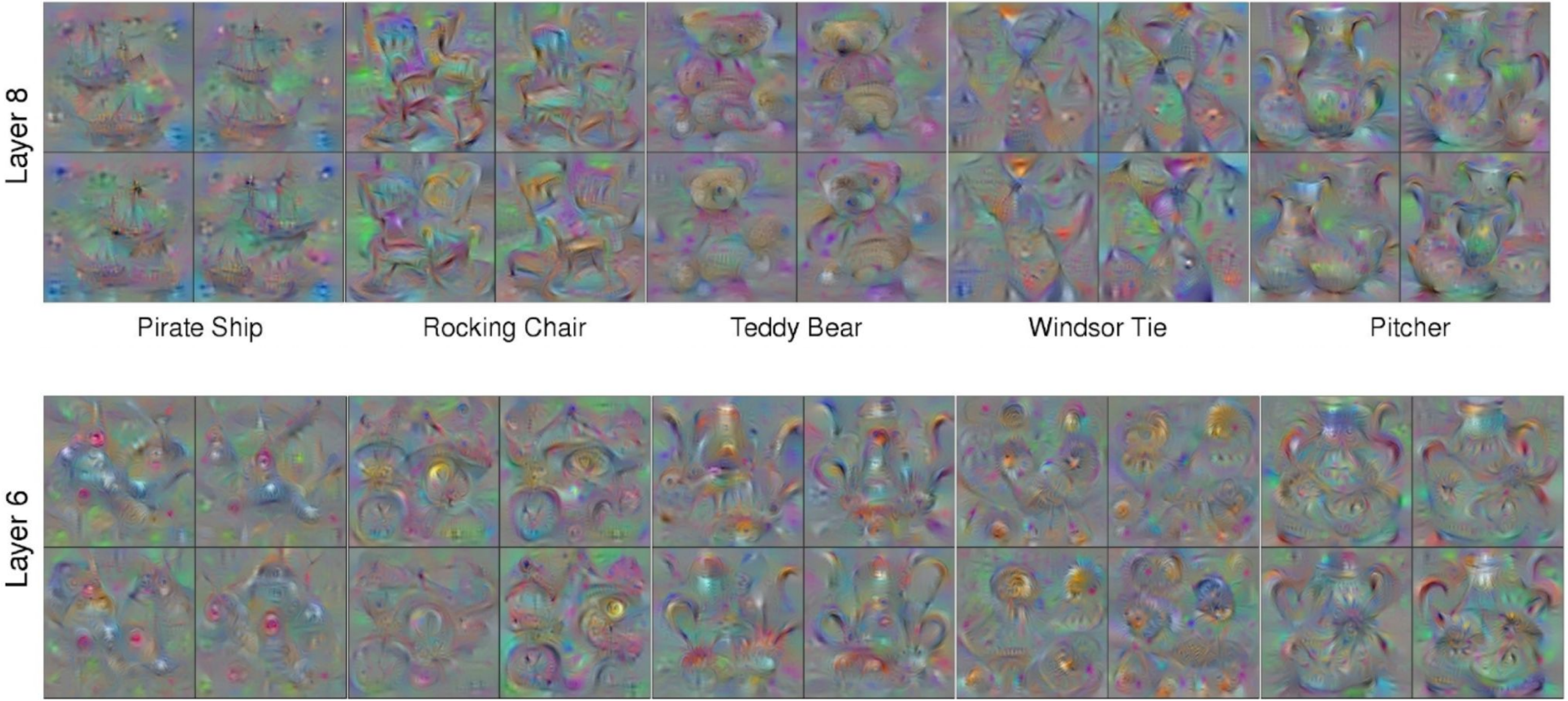
goose | ostrich | limousine

We can in fact do this for arbitrary neurons along the ConvNet



Repeat:
1. Forward an image
2. Set activations in layer of interest to all zero, except for a 1.0 for a neuron of interest
3. Backprop to image
4. Do an "image update"

Higher-layer representations preserve the most important elements of the image



Understanding Convnet - Question

Suppose I have access to an image classifier via an API. I can make a request with an image and receive a response with the predicted probabilities of each class. Which type of analysis would work best in this case?

Understanding Convnet - Question

Suppose I have access to an image classifier via an API. I can make a request with an image and receive a response with the predicted probabilities of each class. Which type of analysis would work best in this case?

Occlusion Experiments

Adversarial Perturbations

How can we fool a CNN?

Given a pre-trained model, modify the input so that it gets misclassified by the CNN but looks identical to the original

$$\max_{x'} \mathcal{L}(x', y) \qquad \text{s.t.} \quad ||x - x'||_\infty \leq \epsilon$$

Maximize loss without allowing input to change by more than some ϵ

Adversarial Perturbations

Maximize loss without allowing input to change by more than some ϵ

The more we change x, the better we can fool the network

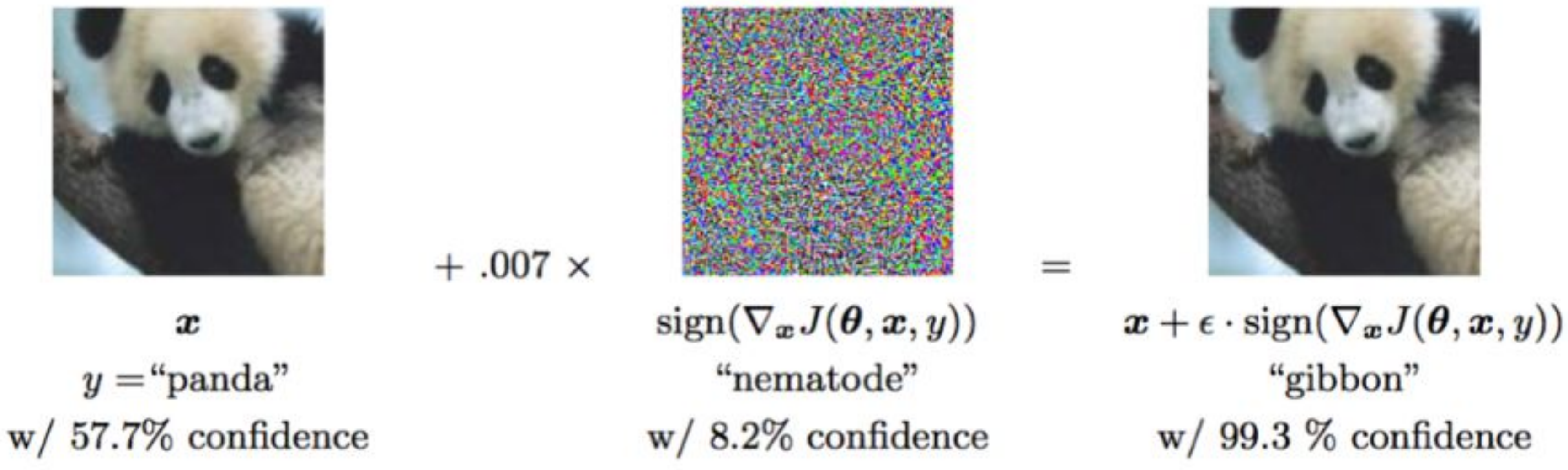
Lets change x as much as possible: every pixel gets +/- ϵ

How do we know whether to increase or decrease?

Use the gradient!

$$x' = x + \epsilon \cdot \mathrm{sign}(\nabla \mathcal{L}(x, y))$$

Adversarial Perturbations



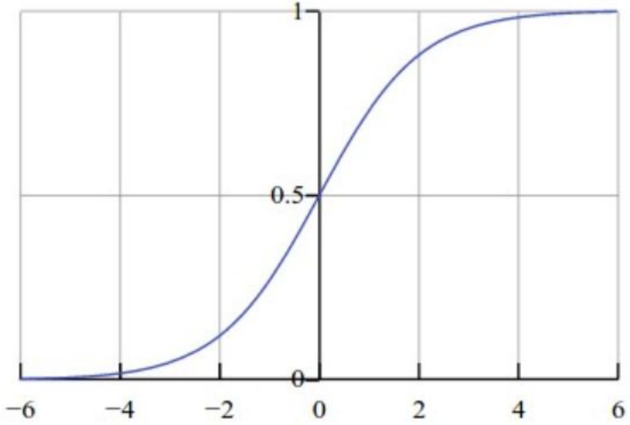
Explaining and Harnessing Adversarial Examples

Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy

Adversarial Perturbations - Example

Lets fool a binary linear classifier:
(logistic regression)

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$



x	1	-2	-2	3	-1	2	-2	-3
w	2	-1	-1	-1	2	-1	-1	1

$\sigma(w^T x+0) = \sigma$
$(2+2+2-3-2-2+2-3) =$
$\sigma(-2) = 0.1192$

Adversarial Perturbations - Example

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

x	1	-2	-2	3	-1	2	-2	-3
w	2	-1	-1	-1	2	-1	-1	1
x’	?	?	?	?	?	?	?	?

How much do we need to change x to fool the classifier?

- Try different epsilon, ϵ = 0.1, 0.2, 0.3
- For each of these, find x’ and determine if it is misclassified
- Hint: the sign of the gradient in this case is the sign of w

Adversarial Perturbations - Example

$\epsilon = 0.1$

x	1	-2	-2	3	-1	2	-2	-3
w	2	-1	-1	-1	2	-1	-1	1
x’	1.1	-2.1	-2.1	2.9	-0.9	1.9	-2.1	-2.9

$\sigma(w^T x' + 0) = \sigma$
$(2.2+2.1+2.1-2.9-1.8-1.9+2.1-2.9) =$
$\sigma(-1) = 0.2689$

Adversarial Perturbations - Example

$\epsilon = 0.3$

x	1	-2	-2	3	-1	2	-2	-3
w	2	-1	-1	-1	2	-1	-1	1
x’	1.3	-2.3	-2.3	2.7	-0.7	1.7	-2.3	-2.7

$\sigma(w^T x' + 0) = \sigma$
$(2.6+2.3+2.3-2.7-1.4-1.7+2.3-2.7) =$
$\sigma(1) = 0.7311$

Adversarial Perturbations - Example

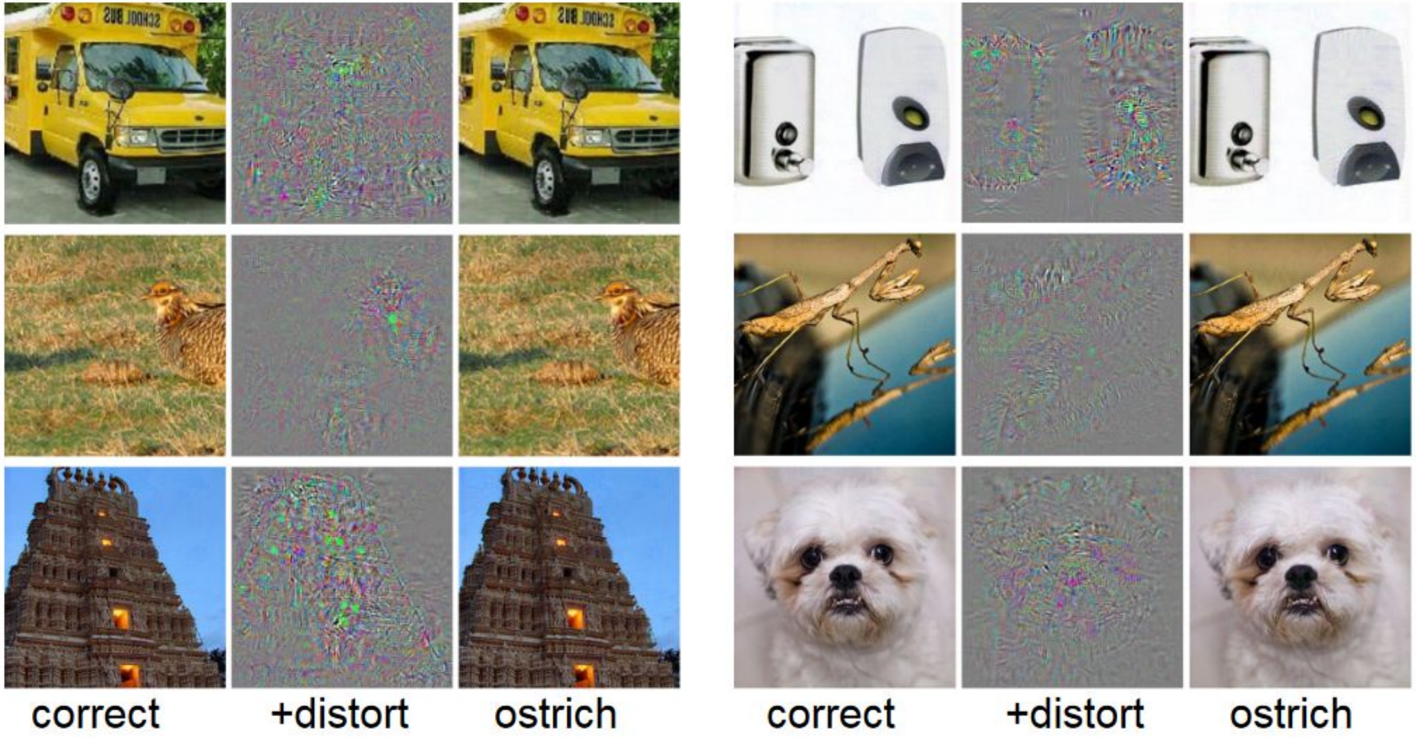
$\epsilon = 0.2$

x	1	-2	-2	3	-1	2	-2	-3
w	2	-1	-1	-1	2	-1	-1	1
x’	1.2	-2.2	-2.2	2.8	-0.8	1.8	-2.2	-2.8

$\sigma(w^T x' + 0) = \sigma$
$(2.4+2.2+2.2-2.8-1.6-1.8+2.2-2.8) =$
$\sigma(0) = 0.5$

Adversarial Training

[Intriguing properties of neural networks, Szegedy et al., 2013]



Style Transfer

Content Image

+

Style Image

=

Style Transfer!

We can use CNNs to transfer the style of one image to another!

- Optimize input image that minimizes style loss and content loss

Style Transfer - Style Loss

For each layer, we compute a Gram Matrix:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

Style Transfer - Style Loss

For each layer, we compute a Gram Matrix:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

1. For a specific pixel, extract the C features and take the outer product, (c,1) x (1,c) = (c,c) matrix

Style Transfer - Style Loss

For each layer, we compute a Gram Matrix:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

1. For a specific pixel, extract the C features and take the outer product, (c,1) x (1,c) = (c,c) matrix
2. Sum across all pixels to produce the final Gram Matrix

Style Transfer - Style Loss

For each layer, we compute a Gram Matrix:

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

1. For a specific pixel, extract the C features and take the outer product, (c,1) x (1,c) = (c,c) matrix
2. Sum across all pixels to produce the final Gram Matrix

Intuitively, G_{ij} means “how much does channel i correlate with channel j”

Style Transfer - Style Loss

Let $\hat{G}^l$ be the Gram Matrix from features of the style image (for a specific layer)

The style loss is defined as:

Style Transfer - Style Loss

Let $\hat{G}^l$ be the Gram Matrix from features of the style image (for a specific layer)

The style loss is defined as:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2$$

"Mean squared error between Gram Matrices"

Style Transfer - Style Loss

Let $\hat{G}^l$ be the Gram Matrix from features of the style image (for a specific layer)

The style loss is defined as:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2$$

"Mean squared error between Gram Matrices"

$$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

"Weighted average across layers"

Style Transfer - Content Loss

We want our output to have the same content as the content image

How about MSE loss on the images?

Style Transfer - Content Loss

We want our output to have the same content as the content image

How about MSE loss on the images? Too restrictive!

Style Transfer - Content Loss

We want our output to have the same content as the content image

How about MSE loss on the images? Too restrictive!

Instead, compute the MSE loss between the CNN features themselves

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2$$

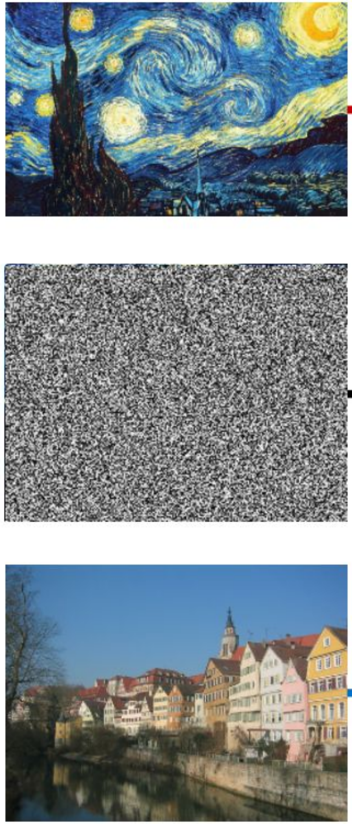
If the features are close, the "content" should be similar

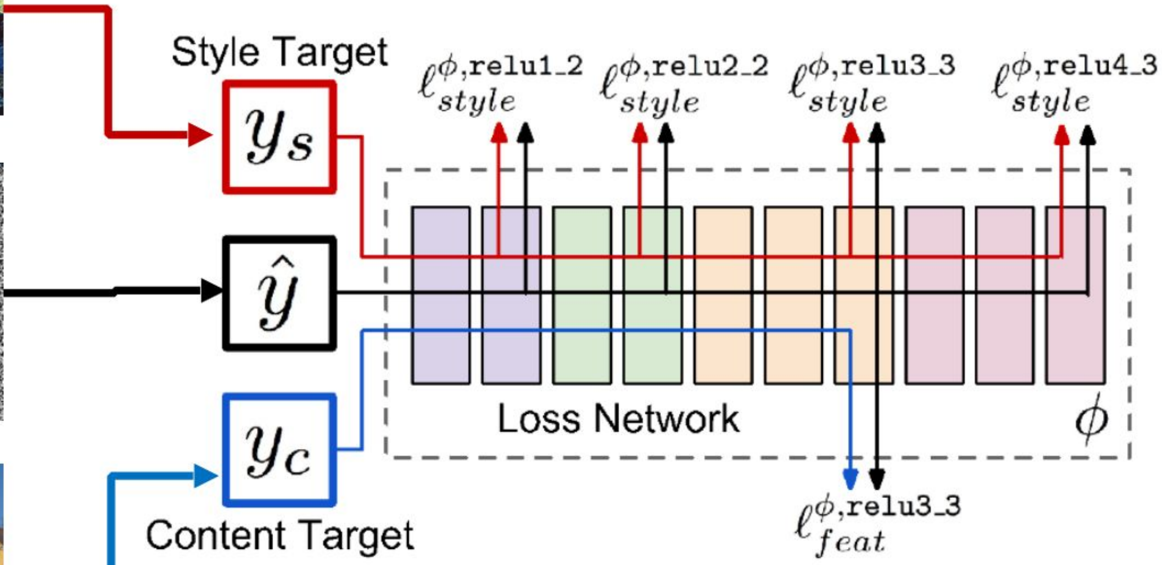
Style Transfer

Style image

Output image (Start with noise)

Content image





Gatys, Ecker, and Bethge, “Image style transfer using convolutional neural networks”, CVPR 2016
Figure adapted from Johnson, Alahi, and Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”, ECCV 2016. Copyright Springer, 2016. Reproduced for educational purposes.

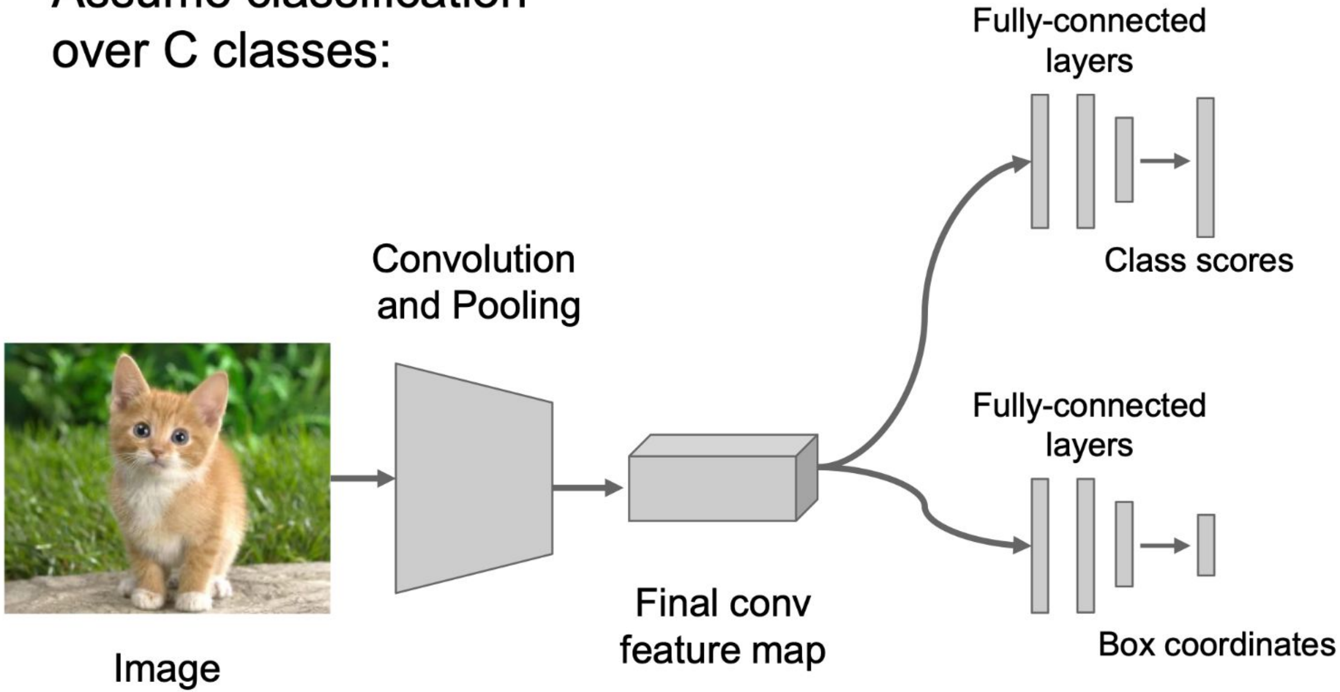
Style Transfer - Question

Why would we want to compute the content loss with a deeper layer of the network?

What happens if we used features from an earlier layer?

Detection using classification

Assume classification over C classes:



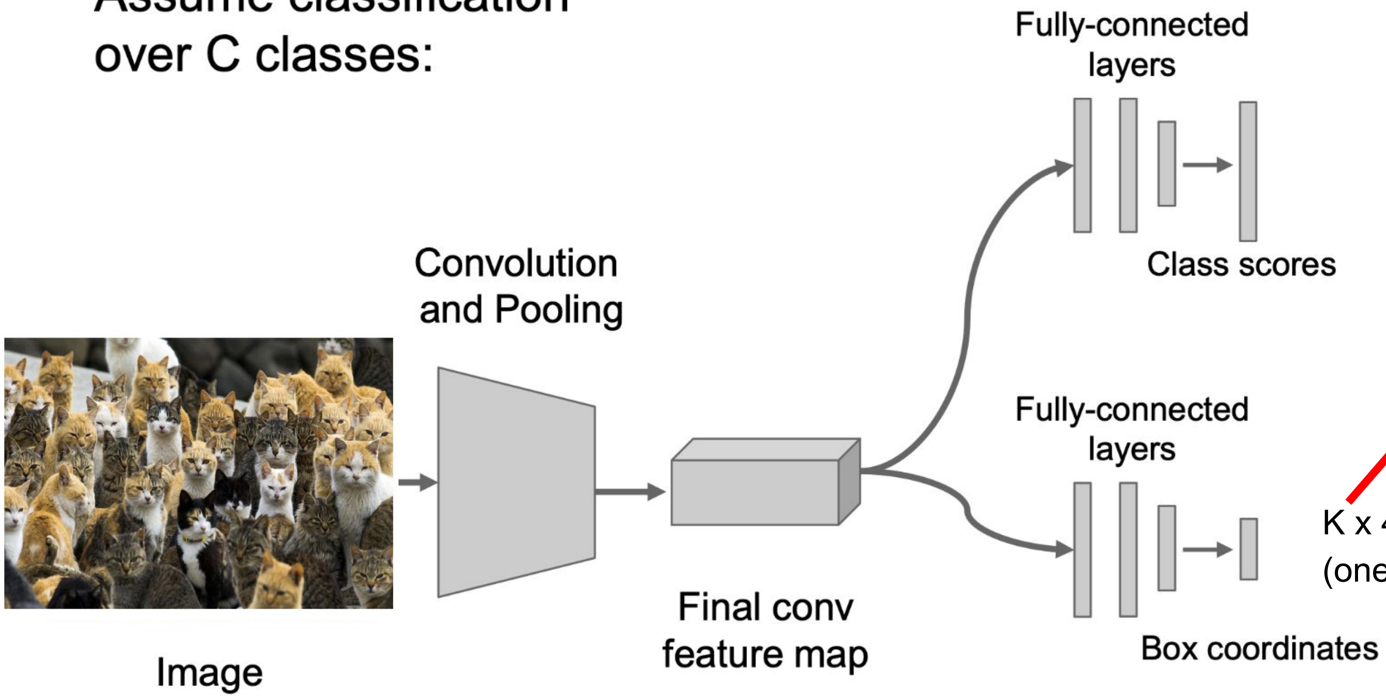
Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

Class specific:
C x 4 numbers
(one box per class)

Detection when multiple objects in the image

Assume classification over C classes:

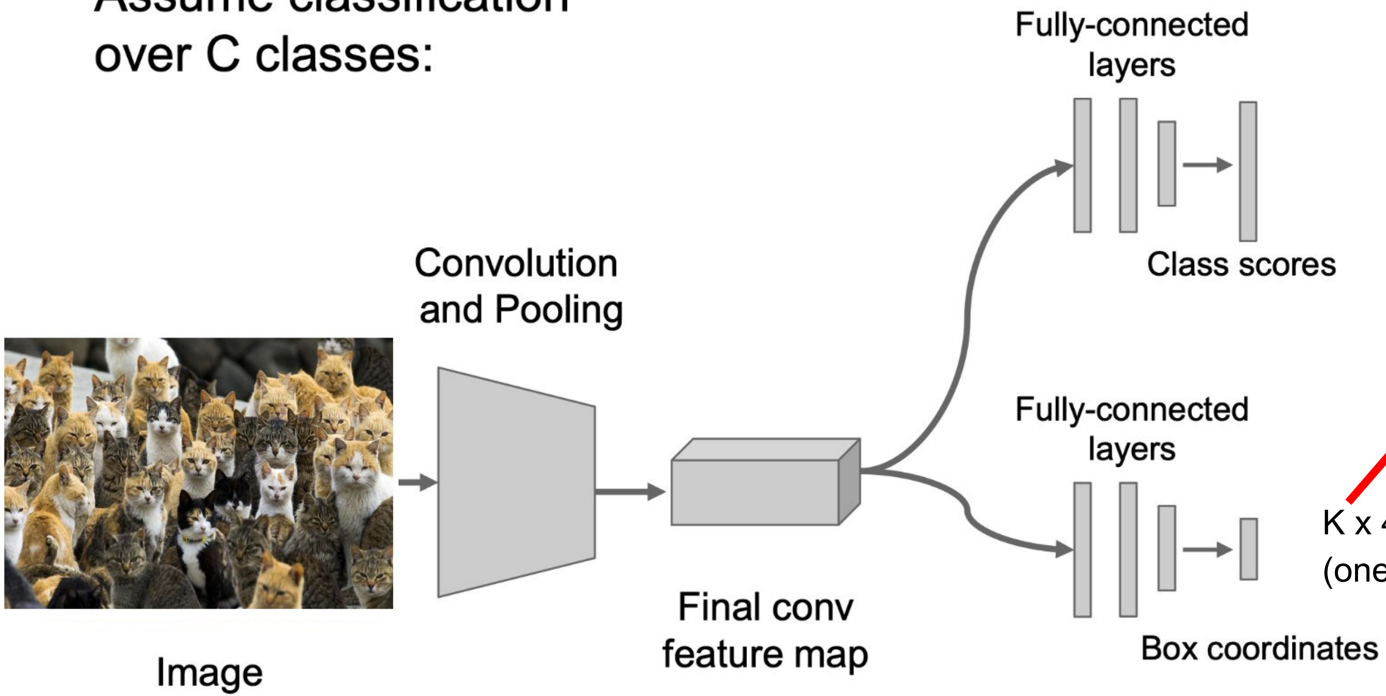


K needs to vary

K x 4 numbers
(one box per object)

Detection when multiple objects in the image

Assume classification over C classes:



K needs to vary

K x 4 numbers
(one box per object)

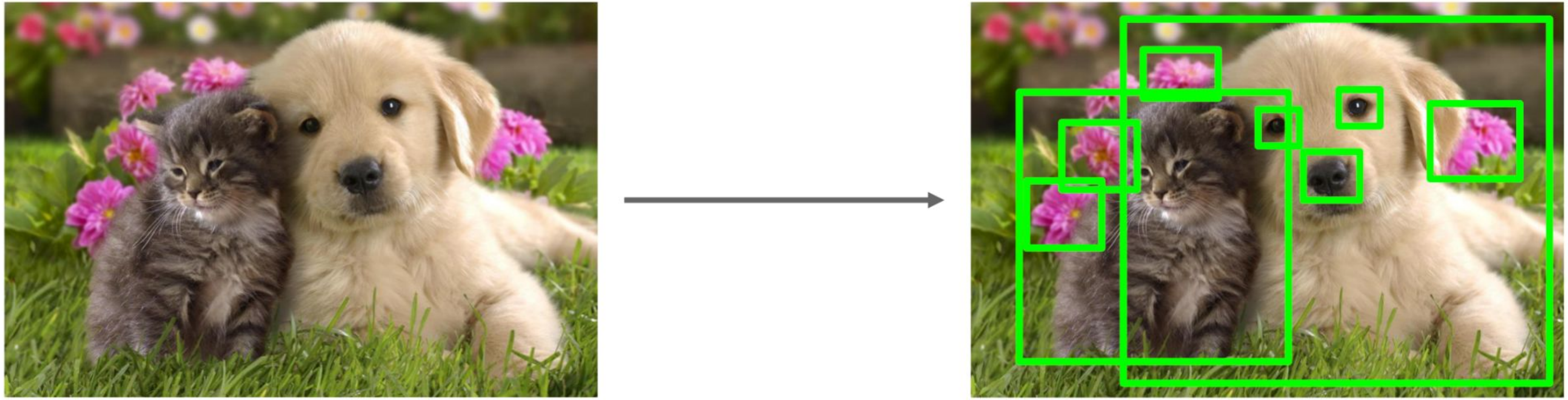
Sliding Window approach

- Run classification + regression network at multiple locations on a high-resolution image
- Convert fully-connected layers into convolutional layers for efficient computation
- Combine classifier and regressor predictions across all scales for final prediction

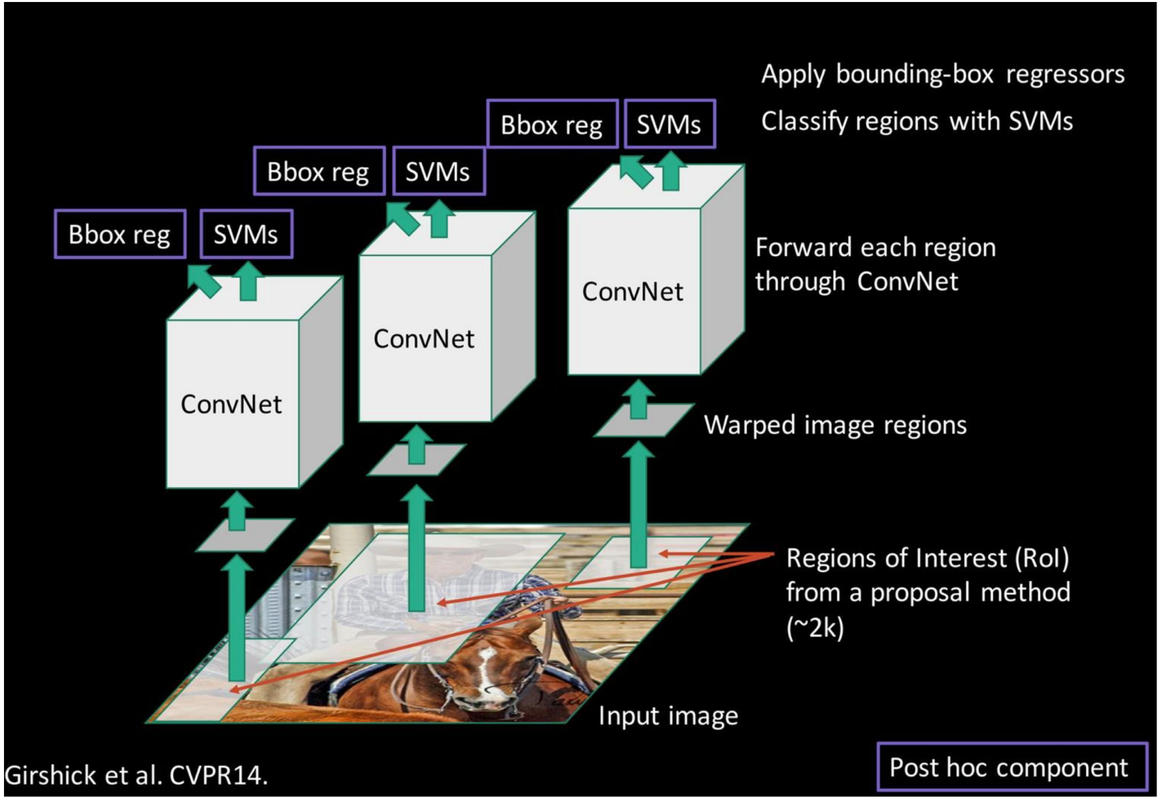
But too many locations and therefore too slow

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Putting it together: R-CNN



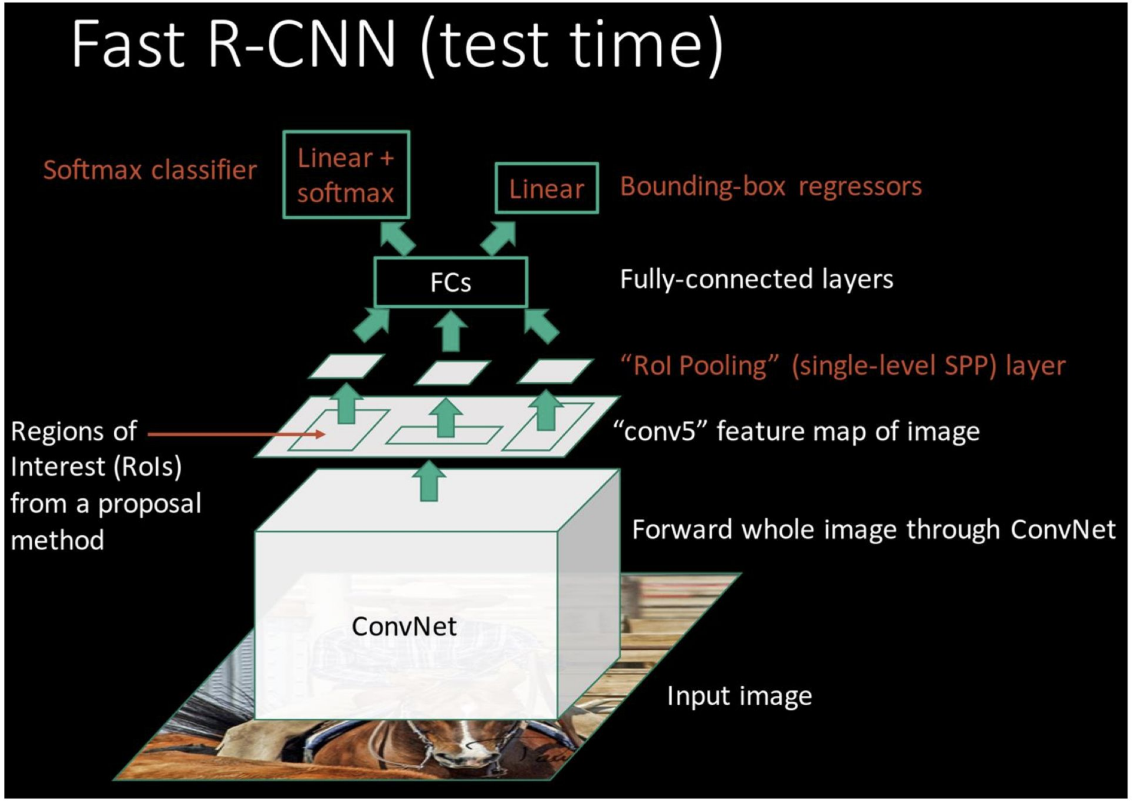
R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal

2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

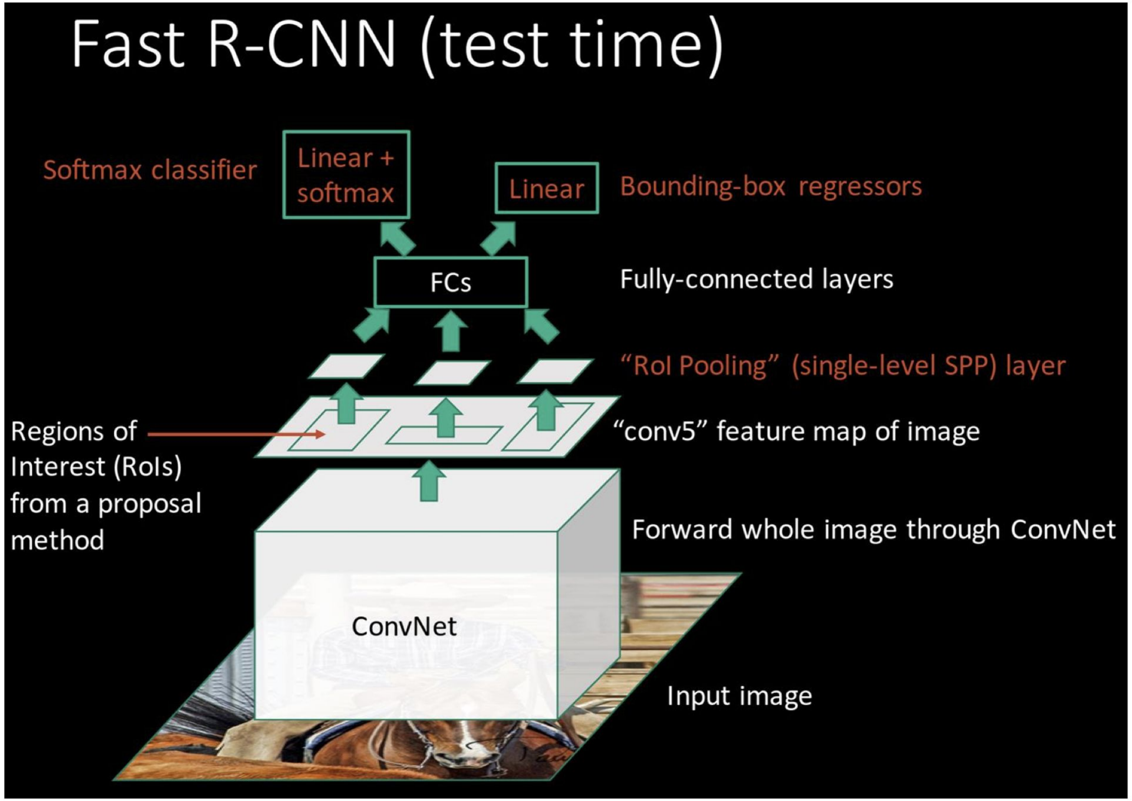
3. Complex multistage training pipeline

Fast RCNN



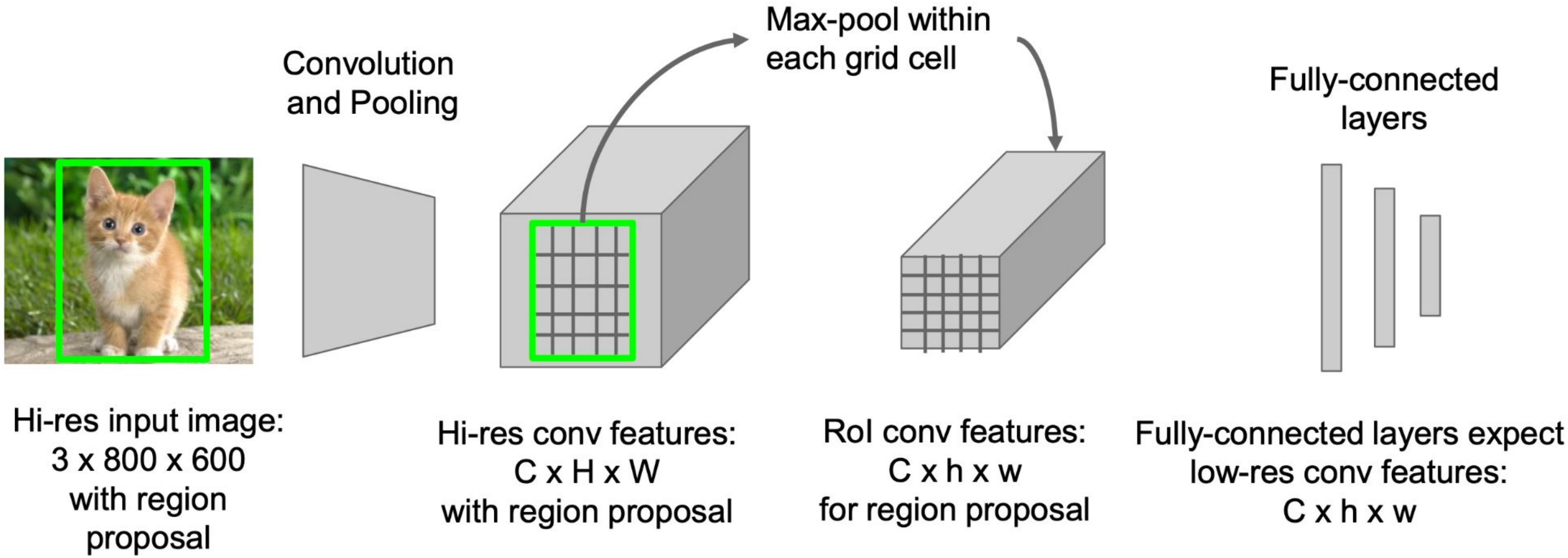
1. Share computation till conv5
2. End-to-end training

Fast RCNN



1. Share computation till conv5
2. End-to-end training

Fast R-CNN: Region of Interest Pooling

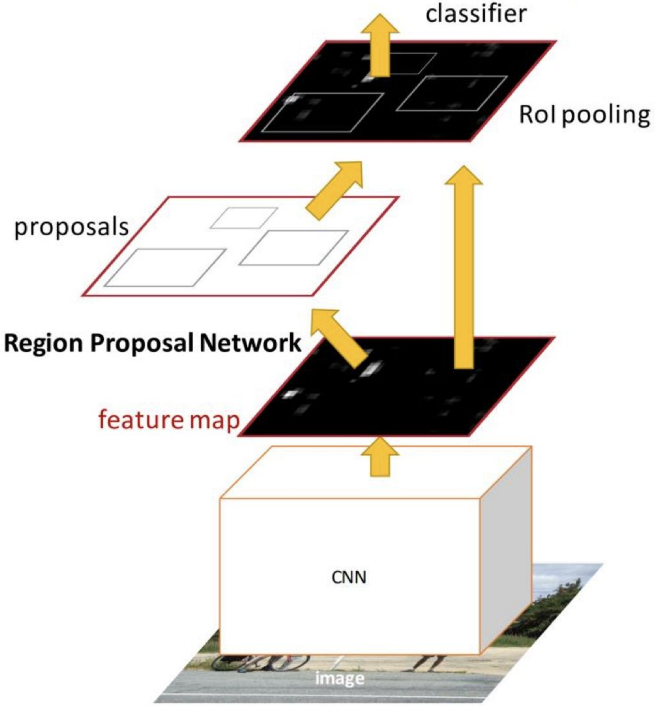


Fast RCNN problem

Region proposal is costly

Make region proposal learnable

Faster R-CNN:



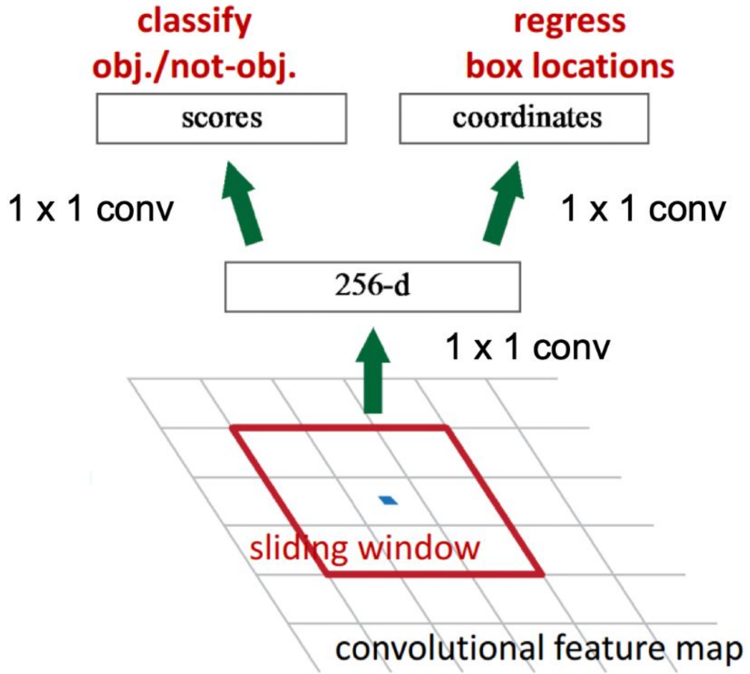
Faster R-CNN: Region Proposal Network

Slide a small window on the feature map

Build a small network for:
- classifying object or not-object, and
- regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



Slide credit: Kaiming He

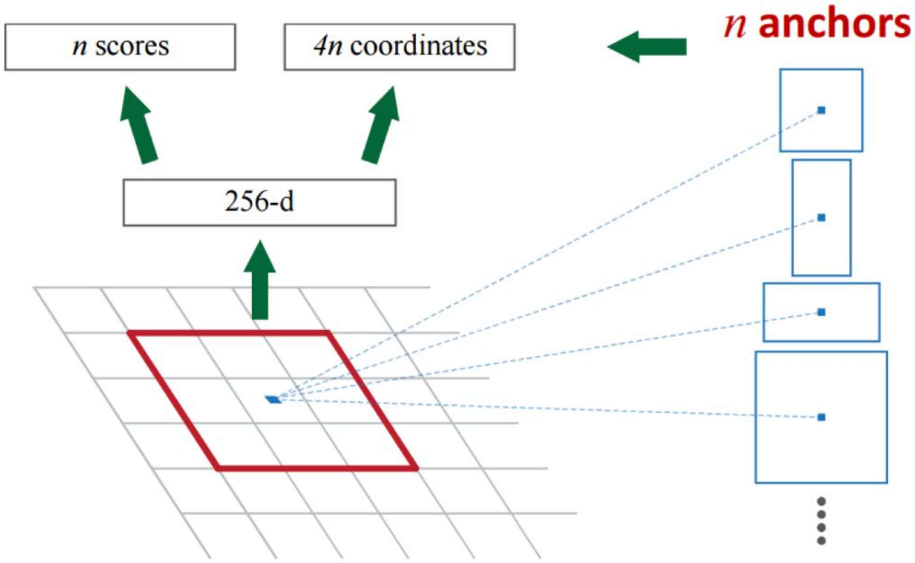
Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object



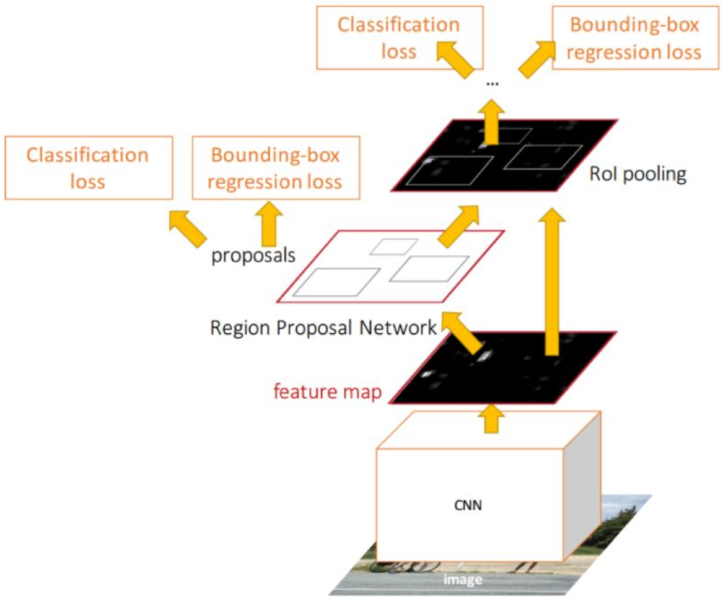
Faster R-CNN: Training

In the paper: Ugly pipeline

- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!
One network, four losses

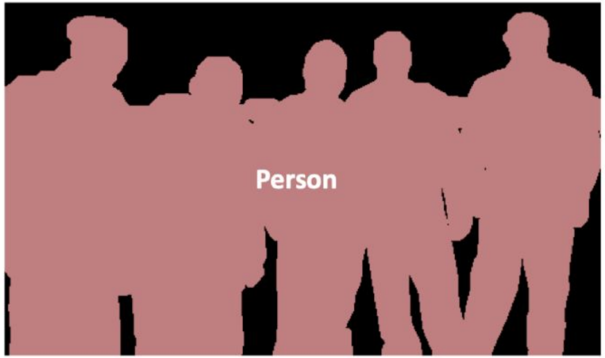
- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)



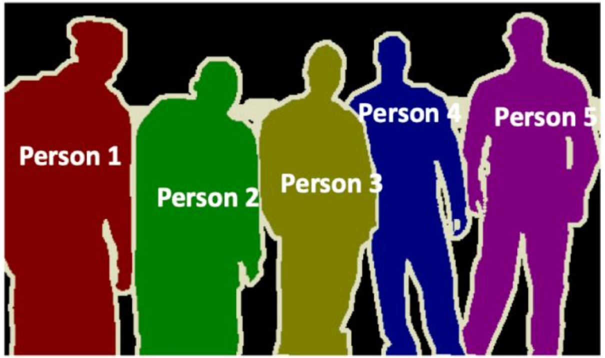
Slide credit: Ross Girschick

Semantic vs Instance Segmentation

Object Detection

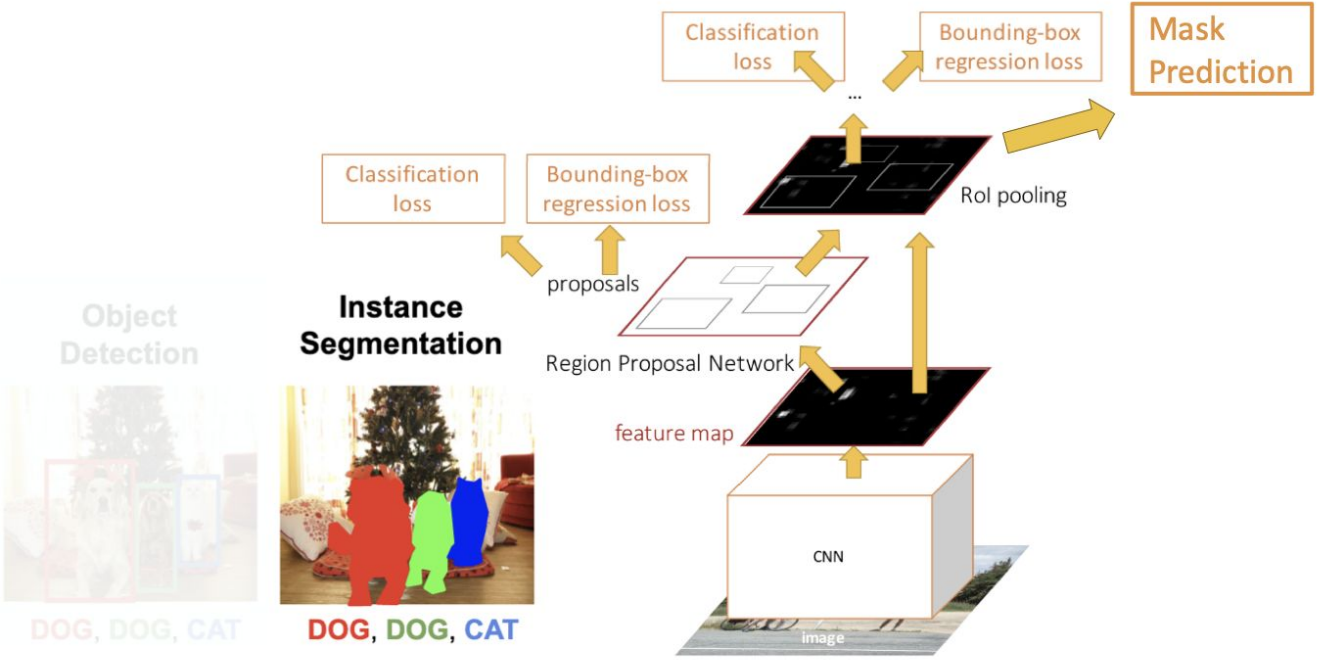


Semantic Segmentation



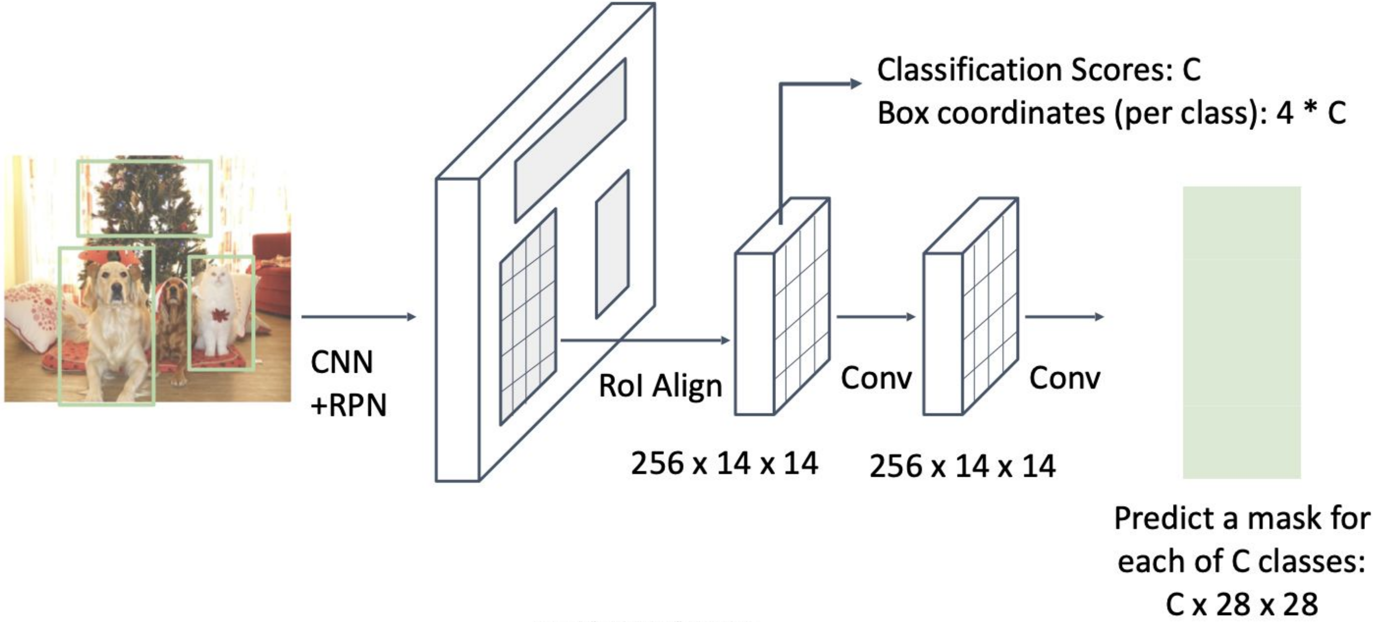
Instance Segmentation

Instance Segmentation: Mask R-CNN



He et al, "Mask R-CNN", ICCV 2017

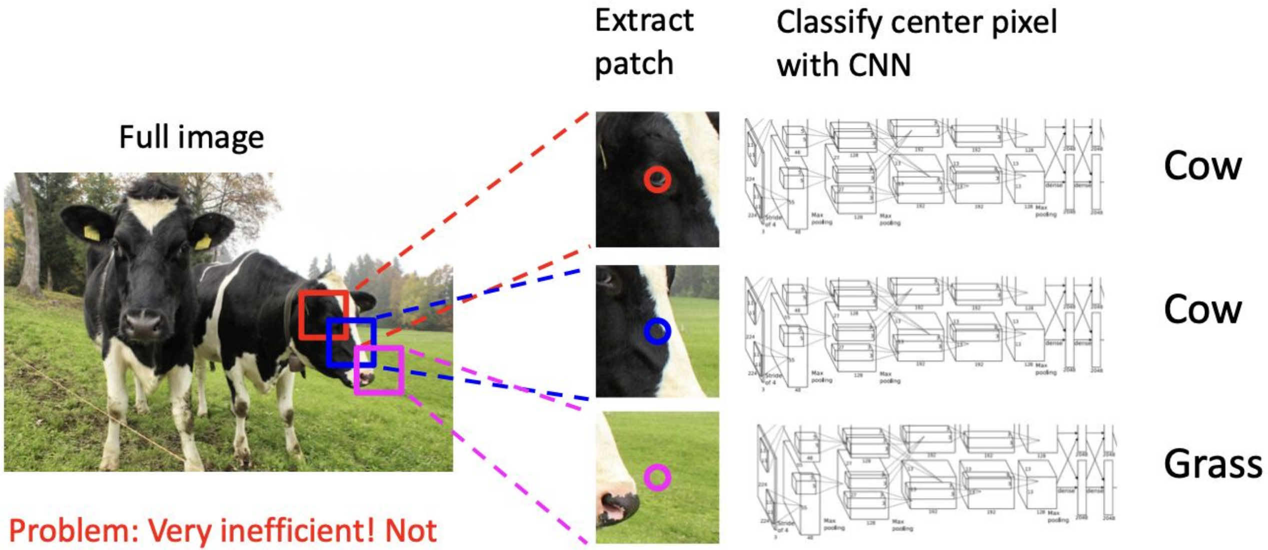
Mask R-CNN



He et al, "Mask R-CNN", ICCV 2017

FCN on ROI

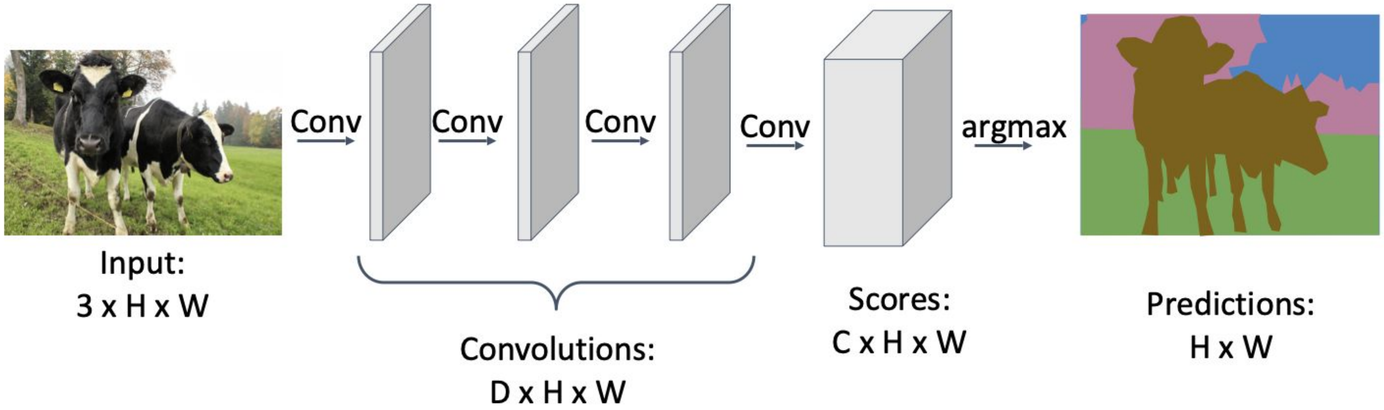
Segmentation: Sliding Window



Problem: Very inefficient! Not reusing shared features between overlapping patches

Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



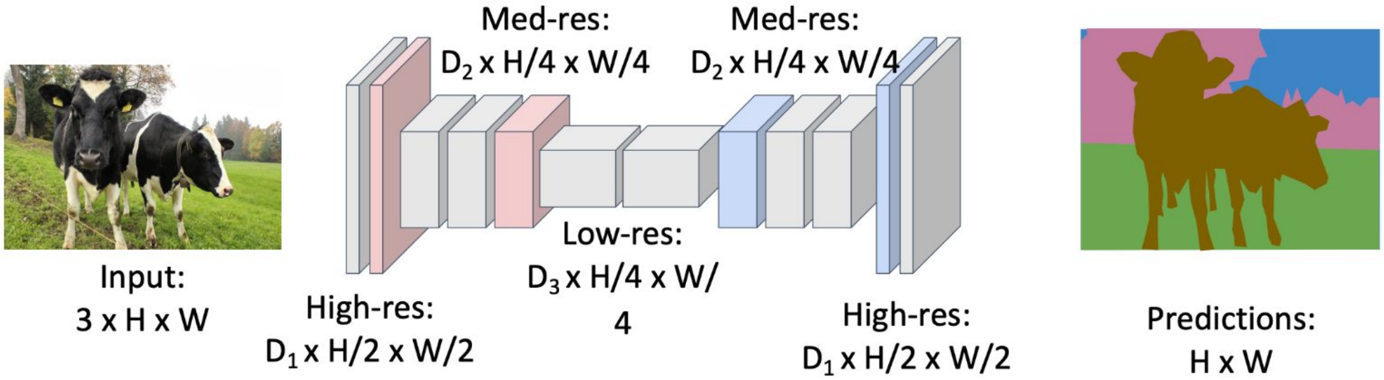
Loss function: Per-Pixel cross-entropy

Problem #1: Effective receptive field size is linear in number of conv layers: With L 3x3 conv layers, receptive field is 1+2L

Problem #2: Convolution on high res images is expensive!

Fully Convolutional Network

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



Downsampling:
Pooling, strided convolution

Upsampling:

Interpolation, ConvTranspose

Spatial Localization - Question

What are some factors to consider when deciding whether to do object detection or segmentation to determine what’s in an image?